Offensive Query Detection

Information Retrieval Midterm Presentation

Problem Statement

The problem is about detecting offensive content in social media posts, and classifying them into different categories such as hateful/ offensive, and clean. The task requires developing an approach to detect and classify the content with high precision and recall, and comparing various models for hate-speech detection.

Additionally, the approach should be able to generalize to different languages without the use of large scale Universal Language Models. The overall objective is to develop a method that can effectively filter out offensive content from social media platforms, to ensure a safe and positive environment for users.

Previous works

- **Convolutional Neural Networks (CNNs)** capture local features in text data and are effective for identifying offensive content, but may not capture long-term dependencies.
- **Recurrent Neural Networks (RNNs)** model temporal dependencies in a sentence and have shown good performance, but may struggle with longer sequences and overfitting.
- **Support Vector Machines (SVMs)** handle large feature sets and sparse data, and have been used for feature selection and classification in detecting offensive language.
- **Google's BERT** uses a pre-training method to generate contextualized word embeddings. It has achieved state-of-the-art performance in detecting offensive language. (F1 score ~0.92)
- **Ensemble models** combine multiple classifiers to improve overall performance.
- **Multi-task learning** allows for training a single model to perform multiple related tasks, which can improve performance in detecting offensive language by leveraging shared features.

To improve models, researchers have explored combining these techniques, using pre-training and transfer learning, and addressing issues related to dataset bias and generalization to different languages.

Classes

- **Contains Curse words and Offensive**
  This category may make up around 5-10% of social media messages, as the use of explicit curse words in an offensive context is generally less common than other forms of communication on social media.

- **Contains curse words but not offensive**
  This category may make up around 20-30% of social media messages, as many users may use curse words for emphasis, humor, or other reasons without intending to be offensive.

- **Do not contain curse words but offensive**
  This category may make up around 10-20% of social media messages, as offensive language or content can be conveyed through other means such as hate speech, discriminatory attitudes, or other forms of negativity.

- **Do not contain curse words and not offensive**
  This category may make up around 50-70% of social media messages, as many users may post neutral or positive content that does not contain any offensive language or intent.

Experiments over English Datasets

Sentiment Analysis

Introduction

Sentiment analysis aids offensive query detection by analyzing query tone and intent to identify and prevent potentially harmful content from being displayed to users.

Uses

By analyzing the sentiment of the words and phrases in a query, sentiment analysis algorithms can detect negative sentiment or tone, which can be used to flag potentially offensive queries.

Context

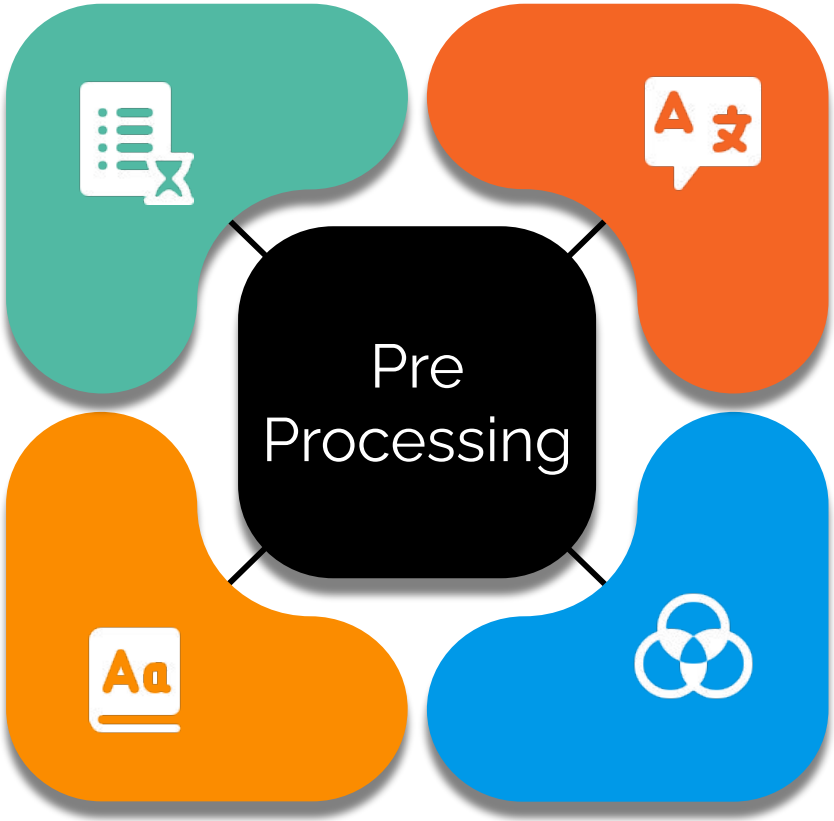
In the context of offensive query detection, sentiment analysis can be used to detect the presence of offensive or inappropriate language, even if explicit curse words are not present.

Porter Stemming

Stemming is a technique used to reduce words to their base or root form.

Stop words

Stop word tokenization involves removing frequently used words such as "the," "and," and "in" from a text.



Tokenization

Tokenization is the process of breaking down a text into smaller units called tokens or words.

Filter Text

Filtering words is used to remove links and emojis from a text, improving its readability and analysis

Naive Bayes Method

- Dataset Collection:
  - We collected the dataset/tweets from various GitHub repositories.
- Dataset Preprocessing:
  - Perform **Tokenization** on dataset to break it into individual words.
  - Remove common **stopwords** from the dataset, such as "the", "and", "is", etc.
  - Applied Porter**Stemmer** and WordNet**Lemmatizer** to reduce words to their root form.
- Train-Test Classifier:
  - Training: Used the preprocessed data to train a Naive Bayes model.
  - Testing: Passed the preprocessed test data through the detection model.
  - Accuracy Evaluation: Noted the accuracy of the model.

Naïve Bayes Model

Data Collection

Collected the dataset/tweets from various GitHub repositories.

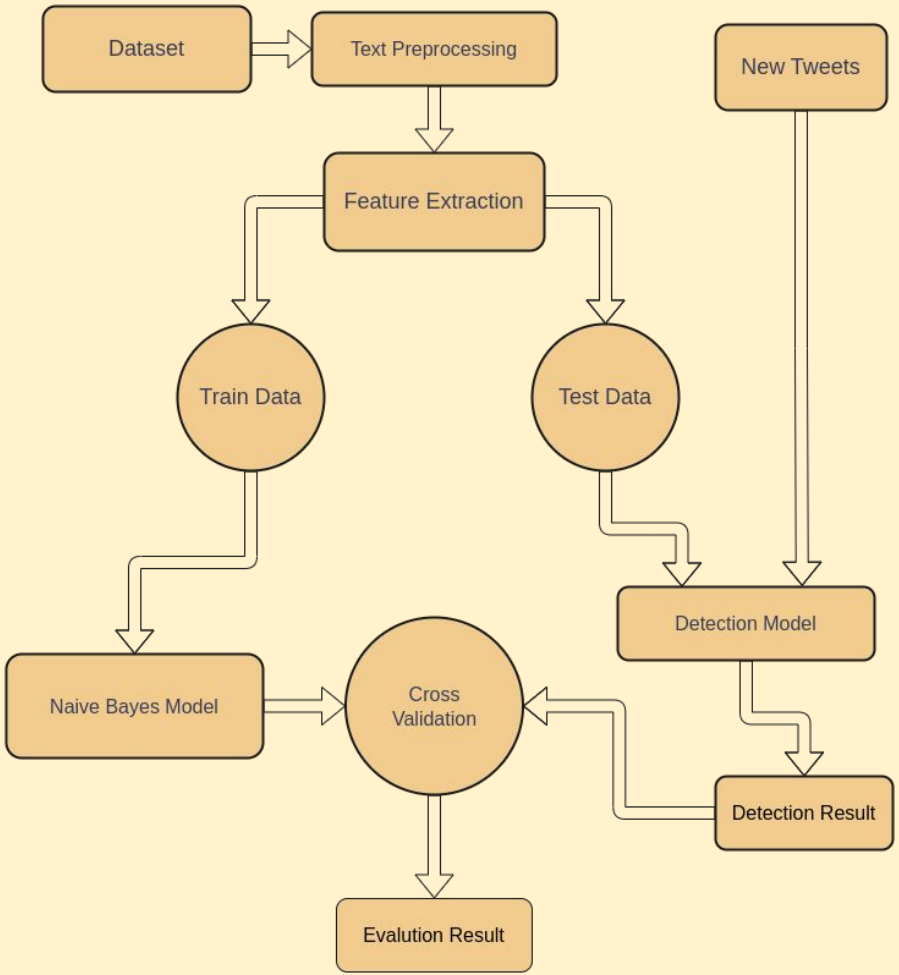
Pre Processing

Perform Tokenization, stopword-removal.
Apply Porter**Stemmer** & WordNet**Lemmatizer.**

Train-Test

Use preprocessed data to train - test the model.
Note metrics - accuracy, precision, recall, F1 Score

Pipeline for our Model



Evaluation Result:

Experiment (score)	Accuracy	Precision	recall	F1 score
TF: 1 + log(tf)	0.8361	0.8353	0.9993	0.9100
TF-IDF: (1 + log(tf)) N/log(df)	0.8407	0.8394	0.9991	0.9123
Bag-of-Words: log((1+tf)/((total_count + len(vocab)))	0.9127	0.9318	0.9653	0.9483
Variation 4: log((1 + df) / (total_doc_count + len(vocab))	0.9071	0.9173	0.9760	0.9457

Experiments over Hindi Datasets

Approaches

Translation

Translate Hindi queries into English using a pre-trained translation model, and then use the English model to detect offensive queries.

This approach is limited by the vocabulary/ correctness of the output generated by the translation model and the accuracy of our translation model.

Parallel Corpus

Train separate offensive query detection models for each language using a parallel corpus of offensive and non-offensive queries.

This approach requires large, labelled parallel corpus for each language.

Gives unsatisfactory result on training over small datasets.

Approaches

Translation

Translate Hindi queries into English using a pre-trained translation model, and then use the English model to detect offensive queries.

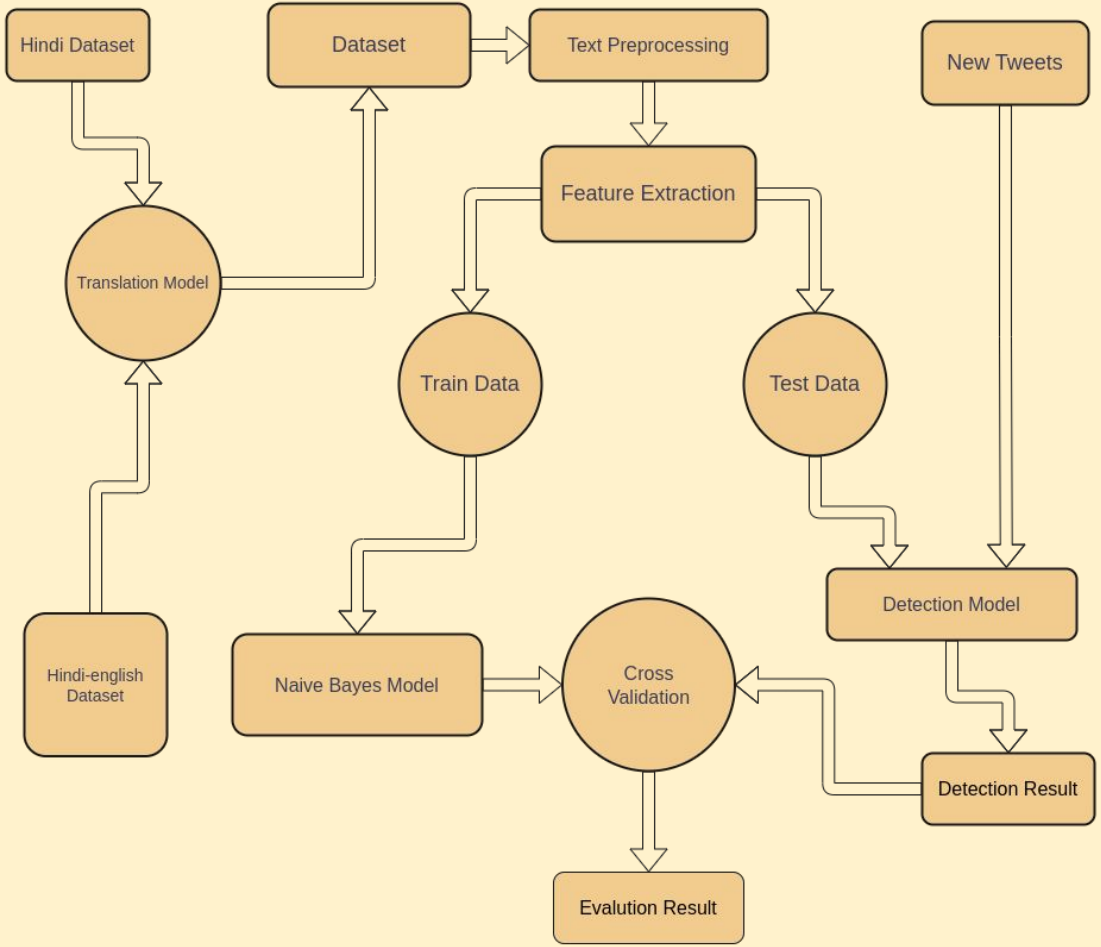
This approach is limited by the vocabulary/ correctness of the output generated by the translation model and the accuracy of our translation model.

Parallel Corpus

Train separate offensive query detection models for each language using a parallel corpus of offensive and non-offensive queries.

This approach requires large, labelled parallel corpus for each language.

Gives unsatisfactory result on training over small datasets.



Pipeline for our Model

Translation Model

- **Using GoogleTrans python module:**
  - Requires internet connection, slow over whole dataset.
  - Output neutralises certain extreme terms.

- **Dictionary based translation:**
  - Bag of words model (Naive Bayes) ignores contextual relations.
  - Can use direct token-based mapping from source language to english.
  - Augment vocabulary through query expansion using synonyms:
    - Increase word-count of all synonymous words during training.
    - Do query expansion using upto four synonyms per word for testing every instance (query).

—

Evaluation Result for Hindi corpus:

Experiment (score)	Accuracy	Precision	recall	F1 score
TF: 1 + log(tf)	0.7888	0.9737	0.1331	0.2342
+ With Bad Word Detection	0.8255	0.7805	0.2602	0.3902
+ More weightage to Hate Comments	0.8272	0.5937	0.6179	0.6056
+ Sentiment Analysis	0.8272	0.5937	0.6178	0.6056

Sources

1) https://github.com/t-davidson/hate-speech-and-offensive-language
2) https://libstore.ugent.be/fulltxt/RUG01/001/887/239/RUG01-001887239_2012_0001_AC.pdf
3) https://sites.google.com/site/offensevalsharedtask/home
4) https://github.com/bdrillard/english-hindi-dictionary/blob/master/English-Hindi%20Dictionary.csv

Team Members

Akash Das	20CS10006
Gaurav Malakar	20CS10029
Roopak Priydarshi	20CS30042
Atishay Jain	20CS30008